CI4251 - Programación Funcional Avanzada
Tareas 5

Ernesto Hernández-Novich
86-17791
<emhn@usb.ve>

Junio 22, 2015

Un problema de programación dinámica...

Considere una expresión booleana compuesta por una secuencia arbitraria de las palabras reservadas:

- `true`
- `false`
- `and`
- `or`
- `xor`

el problema consiste en determinar de *cuántas* maneras se pueden incorporar paréntesis *explícitos* de modo que la expresión tenga el valor `true`.
Por ejemplo, si se nos proveyera le expresión

```
true xor false and true
```

el algoritmo debería contestar `2`, pues esa expresión sólo se hace cierta si se incorporan los paréntesis

```
((true xor false) and true)
(true xor (false and true))
```

Ud. debe implantar en Haskell una solución a este problema utilizando técnicas de programación dinámica apoyadas en arreglos Haskell. En este sentido, construiremos la solución comenzando con un tipo de datos para representar los símbolos involucrados:

```
data Symbol = SymTrue | SymFalse | SymAnd | SymOr | SymXor
            deriving (Show,Eq)
```

Escriba un reconocedor `Parsec` que sea capaz de convertir una expresión construida con los literales, y llevarla a una lista de valores de nuestro tipo algebráico. En este sentido:

- Su reconocedor debe ser capaz de reconocer *varias* expresiones, separadas entre sí por un punto y coma (`;`). Cada expresión puede ocupar una o más líneas, e incluso podría haber más de una expresión en una línea. Pero *todas* terminan con punto y coma.

- Puede haber una cantidad arbitraria de espacios en blanco antes del comienzo de la expresión, entre los literales, antes del punto y coma, y después del punto y coma. Deben ser ignorados.
- Su reconocedor debe rechazar expresiones sintácticamente incorrectas. No es necesario que se recupere de ese error.

Así, la función principal del reconocedor sería

```
expresiones :: Parser [[Symbol]]
```

Escriba entonces la función

```
trueWays :: [Symbol] -> Int
```

que calcule la cantidad de parentizaciones que hacen `true` la expresión.
El programa principal debe recibir un nombre de archivo como argumento de línea de comandos, y si existe, aplicar el reconocedor sobre los contenidos de ese archivo e indicar la cantidad de parentizaciones para cada expresión. Sólo debe mostrar la expresión y la cantidad de parentizaciones, pero *no* necesita mostrar las parentizaciones específicas.
La solución para este algoritmo es directa y emplea técnicas de programación dinámica sobre arreglos *mutables*. Ud. puede presentar una solución utilizando arreglos mutables sobre el monad `ST`, pero sepa que es perfectamente posible hacerlo con arreglos *inmutables* si Ud. escribe *thunks* de manera astuta.

Aprovechando `Arbitrary`

¿Puede escribir una instancia `Arbitrary` que le ayude a generar casos de prueba interesantes?

Referencias

[1] Descripción del problema y solución imperativa

[2] Explicación del problema por Brian Dean (requiere Flash)